9

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/055,483	01/22/2002	Jared W. Stark IV	10559-559001	7484

20985 7590 02/09/2005

FISH & RICHARDSON, PC
12390 EL CAMINO REAL
SAN DIEGO, CA 92130-2081

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	**PAPER NUMBER**
2183	

DATE MAILED: 02/09/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

PTO-90C (Rev. 10/03)

Office Action Summary	**Application No.** 10/055,483	**Applicant(s)** STARK ET AL.	
	Examiner David J. Huisman	**Art Unit** 2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 December 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-46* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-46* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *27 December 2004* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-46 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 12/27/2004.

Claim Rejections - 35 USC § 112

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claim 13 is rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. More specifically, claim 13 recites the limitation "the unexecuted, speculatively ready instructions" in lines 3-4. There is insufficient antecedent basis for this limitation in the claim.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1-4, 7, 10, 12-20, 23-26, 29, 32-34, 38-42, and 46 are rejected under 35 U.S.C. 102(b) as being anticipated by Stark et al., "On Pipelining Dynamic Instruction Scheduling Logic," (as disclosed by applicant and herein referred to as Stark).

7. Referring to claim 1, Stark has taught a processor comprising:

a) a wakeup loop to hold scheduler instructions including unexecuted instructions, and to indicate ready instructions of the unexecuted instructions that may be ready to be executed. See page 59, column 1, paragraphs 1-2. Note that when an instruction is ready to be executed, it is woken up and eventually selected.

b) at least one of the unexecuted instructions to wakeup and notify at least another of the unexecuted instructions to speculatively wakeup before selection of the at least one of the unexecuted instructions is confirmed. See Fig.11 and section 4 and note that a grandparent will broadcast tags which will then speculatively wake up the grandchild instructions (in Fig.11, the OR wakes up the SUB since the XOR and ADD are already completed). Also, an instruction's existence in every stage after the select stage in the Fig.11 pipeline confirms that the instruction was in fact selected. For instance, when the OR is in the execute stage in cycle 3, this confirms that the instruction was selected for execution; otherwise it wouldn't be in the execute stage. And, the SUB is woken up before cycle 3. So, the wakeup happens before selection is confirmed.

c) a select loop to select at least one of the ready instructions for execution. See the first sentence of section 3.3.

8. Referring to claim 2, Stark has taught a processor as described in claim 1. Stark has further taught a collision handling technique. See the second sentence of section 3.3 and note

that if multiple instructions are ready for execution (collision), some scheme is employed to handle the collision and select one of the instructions.

9. Referring to claim 3, Stark has taught a processor as described in claim 2. Stark has further taught that the collision handling technique includes a predict another wakeup technique. See section 4.3 and note that if multiple instructions become ready for execution at the same time, then the older one will be selected. This is consistent with applicant's disclosed PAW technique in that older instructions will execute first.

10. Referring to claim 4, Stark has taught a processor as described in claim 3. Stark has further taught that the predict another wakeup technique includes a PAW vector. See the end of section 4.3 and note that an instruction age (vector) is associated with each instruction. The age is then used to determine which instruction is the oldest so that it is next to execute.

11. Referring to claim 7, Stark has taught a processor as described in claim 1. Stark has further taught that the scheduler instructions include executed instructions. See section 4.5.2, and note that instructions may be re-executed. That is, the instructions in the reservation stations that are to be re-executed are actually executed instructions in that they have been executed previously.

12. Referring to claim 10, Stark has taught a processor as described in claim 1. Stark has further taught that the wakeup loop includes:

a) a wakeup array to hold the scheduler instructions. See Fig.3 and note that waiting instructions are stored in reservation stations (arrays). And the wakeup logic is part of the reservation stations. See the first paragraph on page 59.

b) wakeup logic to indicate the at least one of the unexecuted instructions that may be ready to be selected for execution. See Fig.6 and its description on page 60, and note that when an instruction's sources are ready, the instruction will be woken up and marked as ready.

13. Referring to claim 12, Stark has taught a processor as described in claim 1. Stark has further taught that the wakeup logic is selected from the group consisting of AND/OR array logic, CAM-style logic, and RAM-style logic. See Fig.10 and note the use of AND/OR gates.

14. Referring to claim 13, Stark has taught a processor as described in claim 1. Stark has further taught that the select loop includes select logic to generate a grant vector indicating at least one of the unexecuted, speculatively ready instructions is granted execution. See Fig.10 and section 4.2 and the first paragraph of section 4.3, and note that when an instruction is speculatively ready and granted execution, a destination tag (vector) is generated and transmitted on the destination tag bus.

15. Referring to claim 14, Stark has taught a processor as described in claim 1. Stark has further taught that the wakeup loop is pipelined over at least two cycles. See sections 3.5. and 4.

16. Referring to claim 15, Stark has taught a processor comprising a select-free scheduler including wakeup and select logic having a total scheduling latency, to schedule instructions for functional units that execute instructions having an execution latency that is less than the total scheduling latency of the wakeup and select logic. See section 3.5 and note that the wakeup and select logic (scheduling logic) requires two cycles because it is pipelined. Furthermore, from Fig.8, it can be seen that the ADD and OR instructions require one execution cycle. Therefore, the execution latency of the functional units is less than the total latency required for scheduling.

17. Referring to claim 16, Stark has taught a processor as described in claim 15. Stark has further taught a dynamic instruction scheduler to execute instructions that have an execution latency that is at least equal to the total execution latency of the wakeup and select logic. Recall from section 3.5 that scheduling requires two cycles. And, Fig.8 shows that a functional unit takes three cycles to execute a load instruction. Therefore, the functional unit latency is at least equal to the latency of the scheduling logic.

18. Referring to claim 17, Stark has taught a processor as described in claim 15. Stark has further taught that the select-free scheduler includes: a wakeup stage to hold scheduler instructions including unexecuted instructions (see the first paragraph on page 59), the wakeup stage including:

a) a wakeup array having resource vectors corresponding to the unexecuted instructions to indicate dependencies upon resources. See Fig.10 and note the resource vector which includes information regarding when an instruction's parents and/or grandparents are ready.

b) wakeup logic to indicate at least one of the unexecuted instructions that may be ready to be selected for execution. See Fig.10 and the first full paragraph in column 2 on page 63. Note that when an instruction is speculatively ready a request is made for execution.

c) select logic including speculative wakeup to indicate at least one of the unexecuted instructions that may be ready to be selected for execution. See Fig.10 and note the select logic, and also see the first few sentences of section 4.3. The logic indicates that the instruction may be ready to execute but a confirm signal is also required which indicates it is ready to execute.

19. Referring to claim 18, Stark has taught a processor as described in claim 17. Stark has taught that the processor further comprises a collision handling technique. See the second

sentence of section 3.3 and note that if multiple instructions are ready for execution (collision), some scheme is employed to handle the collision and select one of the instructions.

20. Referring to claim 19, Stark has taught a processor as described in claim 18. Stark has further taught that the collision handling technique includes a predict another wakeup technique. See section 4.3 and note that if multiple instructions become ready for execution at the same time, then the older one will be selected. This is consistent with applicant's disclosed PAW technique in that older instructions will execute first.

21. Referring to claim 20, Stark has taught a processor as described in claim 19. Stark has further taught that the predict another wakeup technique includes a PAW vector. See the end of section 4.3 and note that an instruction age (vector) is associated with each instruction. The age is then used to determine which instruction is the oldest so that it is next to execute.

22. Referring to claim 23, Stark has taught a system comprising a processor including:

a) a random access memory (RAM) device. It is inherent that a RAM device exists within Stark. This is clear because Stark discloses load instructions (see Fig.8). Loads inherently access RAM if they miss a cache, and Stark discloses cache misses. See the first paragraph of column 2 on page 58.

b) a processor in communication with the RAM device, the processor including:

b1) a wakeup loop to hold scheduler instructions including unexecuted instructions, and to indicate ready instructions of the unexecuted instructions that may be ready to be executed. See Fig.10, section 4.2, and the first paragraph on page 59.

b2) at least one of the unexecuted instructions to wakeup and notify at least another of the unexecuted instructions to speculatively wakeup before selection of the at least one of the

unexecuted instructions is confirmed. See Fig.11 and section 4 and note that a grandparent will broadcast tags which will then speculatively wake up the grandchild instructions (in Fig.11, the OR wakes up the SUB since the XOR and ADD are already completed). Also, an instruction's existence in every stage after the select stage in the Fig.11 pipeline confirms that the instruction was in fact selected. For instance, when the OR is in the execute stage in cycle 3, this confirms that the instruction was selected for execution. And, the SUB is woken up before cycle 3. So, the wakeup happens before selection is confirmed.

b3) a select loop to select at least one of the ready instructions for execution. See section 4.3 and note that an instruction is selected for execution when its parents are ready.

23. Referring to claim 24, Stark has taught a system as described in claim 23. Stark has taught that the system further comprises a collision handling technique. See the second sentence of section 3.3 and note that if multiple instructions are ready for execution (collision), some scheme is employed to handle the collision and select one of the instructions.

24. Referring to claim 25, Stark has taught a processor as described in claim 24. Stark has further taught that the collision handling technique includes a predict another wakeup technique. See section 4.3 and note that if multiple instructions become ready for execution at the same time, then the older one will be selected. This is consistent with applicant's disclosed PAW technique in that older instructions will execute first.

25. Referring to claim 26, Stark has taught a processor as described in claim 25. Stark has further taught that the predict another wakeup technique includes a PAW vector. See the end of

section 4.3 and note that an instruction age (vector) is associated with each instruction. The age is then used to determine which instruction is the oldest so that it is next to execute.

26. Referring to claim 29, Stark has taught a system as described in claim 23. Stark has further taught that the scheduler instructions include executed instructions. See section 4.5.2, and note that instructions may be re-executed. That is, the instructions in the reservation stations that are to be re-executed are actually executed instructions in that they have been executed previously.

27. Referring to claim 32, Stark has taught a method of issuing requesting instructions to an execution unit, comprising:

a) speculatively setting an indicator to indicate a requesting instruction is ready to be selected for execution, said speculatively setting being caused by a prior wakeup of an earlier instruction before selection of the earlier instruction is confirmed, the requesting instruction being a dependent of the earlier instruction. See Fig.10 and sections 4.2 and 4.3. Note that if an instruction's grandparents are ready, for instance, then the instruction wakes up speculatively, by speculatively setting a request signal. Also, see Fig.11 and section 4 and note that a grandparent will broadcast tags which will then speculatively wake up the grandchild instructions (in Fig.11, the OR wakes up the SUB since the XOR and ADD are already completed). It should be noted that an instruction's existence in every stage after the select stage in the Fig.11 pipeline confirms that the instruction was in fact selected. For instance, when the OR is in the execute stage in cycle 3, this confirms that the instruction was selected for execution. And, the SUB is woken up before cycle 3. So, the setting (wakeup) happens before selection is confirmed.

b) during a cycle, selecting a predetermined number of the requesting instructions having a set indicator. See the first paragraph of section 4.3 and section 3.3. Note that one ready instruction is selected for execution based on some heuristic. For instance, as section 4.3 discloses, request priority is one factor that may be taken into consideration when choosing a ready instruction for execution.

c) resetting the indicator of the requesting instructions that are selected. See the last sentence in section 3.3, and note that this is inherent. If an indicator is used to say the instruction is ready, then it must be cleared when the instruction is finally selected for execution. Otherwise, the executed instruction will continue to appear to be ready, and possibly be selected over and over again for redundant execution.

28. Referring to claim 33, Stark has taught a method as described in claim 32. Stark has further taught that the predetermined number of selected instructions is one. See section 3.3 and 4.3.

29. Referring to claim 34, Stark has taught a method as described in claim 32. Stark has taught that the method further comprises handling collisions. See the second sentence of section 3.3 and note that if multiple instructions are ready for execution (collision), some scheme is employed to handle the collision and select one of the instructions.

30. Referring to claim 38, Stark has taught a method as described in claim 34. Stark has taught delaying setting the indicator based on predicting wakeup of another instruction. See section 4.3 and note that if multiple instructions are predicted to wakeup, then the older one will be selected. Therefore, the indication will not be set for the younger instruction so that it will not be selected for execution before the older instruction.

31. Referring to claim 39, Stark has taught a method of issuing unexecuted instructions to an execution unit, comprising:

a) generating resource vectors corresponding to the unexecuted instructions, the resource vectors including resource indicators to indicate availability of resources. See Fig.10 and note the resource vector which includes information regarding when an instruction's parents and/or grandparents are ready. In essence, if the parents or grandparents are ready, it means that the resources on which the instruction is dependent on will be available soon also (as they are produced by the grandparents and parents), and therefore, a resource indicator (request line) is used to make such an indication.

b) speculatively setting the resource indicators to indicate resources associated with corresponding ones of the unexecuted instructions are available so that the corresponding ones of the unexecuted instructions are ready to be executed, said speculatively setting being caused by a prior wakeup of one or more earlier instructions before selection of the one or more earlier instructions is confirmed. See the second paragraph in section 4. Also, from Fig.10, note that the request line is set which indicates that the resources are likely to be available soon and therefore, the instruction should wake up. This setting is caused when the grandparent's of the given instruction are ready. That is, from Fig.11 and section 4, note that a grandparent will broadcast tags which will then speculatively wake up the grandchild instructions (in Fig.11, the OR wakes up the SUB since the XOR and ADD are already completed). It should be noted that an instruction's existence in every stage after the select stage in the Fig.11 pipeline confirms that the instruction was in fact selected. For instance, when the OR is in the execute stage in cycle 3, this confirms that the instruction was selected for execution. And, the SUB is woken up before

cycle 3. So, the wakeup happens before selection is confirmed. Note that selection would also be confirmed in a write/retire/commit stage.

c) selecting a predetermined number of the corresponding ones of the unexecuted instructions. See the first paragraph of section 4.3 and section 3.3. Note that one ready instruction is selected for execution based on some heuristic. For instance, as section 4.3 discloses, request priority is one factor that may be taken into consideration when choosing a ready instruction for execution.

32. Referring to claim 40, Stark has taught a method as described in claim 39. Stark has further taught resetting the resource indicators corresponding to the unexecuted instructions that are selected. See the last sentence in section 3.3, and note that this is inherent. If an indicator is used to say the instruction is ready, then it must be cleared when the instruction is finally selected for execution. Otherwise, the executed instruction will continue to appear to be ready, and possibly be selected over and over again for redundant execution.

33. Referring to claim 41, Stark has taught a method as described in claim 39. Stark has further taught that the predetermined number of selected instructions is one. See section 3.3 and 4.3.

34. Referring to claim 42, Stark has taught a method as described in claim 39. Stark has further taught handling collisions. See the second sentence of section 3.3 and note that if multiple instructions are ready for execution (collision), some scheme is employed to handle the collision and select one of the instructions.

35. Referring to claim 46, Stark has taught a method as described in claim 42. Stark has taught delaying setting the resource indicators based on predicting wakeup of another instruction. See section 4.3 and note that if multiple instructions are predicted to wakeup, then the older one

will be selected. Therefore, the indicators (request and confirm lines) will not be set for the younger instruction so that it will not be selected for execution before the older instruction.

Claim Rejections - 35 USC § 103

36. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

37. Claims 5-6, 8-9, 11, 21-22, 27-28, 30-31, 35-37, and 43-45 are rejected under 35 U.S.C. 103(a) as being unpatentable over Stark, as applied above, in view of Hennessy and Patterson, "Computer Architecture - A Quantitative Approach, 2nd Edition," 1996 (herein referred to as Hennessy).

38. Referring to claim 5, Stark has taught a processor as described in claim 2. Stark has not taught that the collision handling technique includes a scoreboard to indicate whether dependent instructions of an executed instruction have executed. However, Hennessy has taught such a concept. See page 248 and note that the scoreboard indicates that the MULTD, SUBD, and ADDD instructions have all finished completion, and these instructions are dependent instructions of instructions that have already executed. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard.

39. Referring to claim 6, Stark in view of Hennessy has taught a processor as described in claim 5. Hennessy has further taught a pileup victims vector is computed based on the scoreboard. See page 247 and note from the "functional unit status" table that some instructions are not ready to be executed and so they are piling up in the scoreboard. These instructions include all of the instructions with Rj or Rk = No. This means that their operands are not ready and therefore the instructions are just waiting. They are detected by the values of Rj and Rk, which would likely be a value of 0 (for No) and 1 (for Yes).

40. Referring to claim 8, Stark has taught a processor as described in claim 1. Stark has not taught that the select loop generates a collision victim vector to identify collision victims. However, Hennessy has taught the concept of generating a vector within a scoreboard which is used to determine a collision has occurred. See page 247 and note that the SUBD and ADDD instructions collide because they both require the same functional unit (see caption). Since the SUBD is older and also the parent of the ADDD, it will be issued first, thereby generating the vector associated with the SUBD instruction. The vector comprises bits which represent the fields shown in the "functional unit status" and "instruction status" table. In this situation, the important data is that which specifies the ADD unit (which is used to execute the SUBD and ADDD instructions) is busy. By saying it is busy, the ADDD cannot wakeup and be issued for execution (as shown in the "instruction status" table). The bits used to specify the ADDD instruction has not issued indicates that it has collided with another instruction. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. See page 242. Consequently, less stalling is required. As a result it

would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard which is used to detect collision victims.

41. Referring to claim 9, Stark in view of Hennessy has taught a processor as described in claim 8. Hennessy has further taught that the collision victim vector is communicated to the wakeup loop. Again, see page 247. More specifically, this vector is used to direct wakeup of instructions. For instance, when the SUBD instruction is done with the ADD unit, then it will no longer be busy and therefore, the ADDD instruction may be woken up and issued.

42. Referring to claim 11, Stark has taught a processor as described in claim 10.

a) Stark has further taught that the wakeup array includes a resource vector corresponding to each of the scheduler instructions to indicate dependencies upon resources. See Fig.10 and note the resource vector which includes information regarding when an instruction's parents and/or grandparents are ready.

b) Stark has not taught a PAW vector to indicate the resources needed by earlier instructions in the wakeup array. However, Hennessy has taught the concept of bits used to indicate resources needed by earlier instructions. See page 247 and note the "functional unit status" table. Note that bits are used to indicate what functional units are busy and what resources are required by instructions. This type of information allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. See page 242. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to provide a PAW vector for monitoring resources of earlier instructions.

43. Referring to claim 21, Stark has taught a processor as described in claim 18. Stark has not taught that the collision handling technique includes a scoreboard to indicate whether dependent instructions of an executed instruction have executed. However, Hennessy has taught such a concept. See page 248 and note that the scoreboard indicates that the MULTD, SUBD, and ADDD instructions have all finished completion, and these instructions are dependent instructions of instructions that have already executed. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard.

44. Referring to claim 22, Stark in view of Hennessy has taught a processor as described in claim 21. Hennessy has further taught a pileup victims vector is computed based on the scoreboard. See page 247 and note from the "functional unit status" table that some instructions are not ready to be executed and so they are piling up in the scoreboard. These instructions include all of the instructions with Rj or Rk = No. This means that their operands are not ready and therefore the instructions are just waiting. They are detected by the values of Rj and Rk, which would likely be a value of 0 (for No) and 1 (for Yes).

45. Referring to claim 27, Stark has taught a processor as described in claim 24. Stark has not taught that the collision handling technique includes a scoreboard to indicate whether dependent instructions of an executed instruction have executed. However, Hennessy has taught such a concept. See page 248 and note that the scoreboard indicates that the MULTD, SUBD, and ADDD instructions have all finished completion, and these instructions are dependent instructions of instructions that have already executed. A scoreboard allows instructions to

execute out-of-order which allows instructions to execute as soon as their operands are available. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard.

46. Referring to claim 28, Stark in view of Hennessy has taught a processor as described in claim 27. Hennessy has further taught a pileup victims vector is computed based on the scoreboard. See page 247 and note from the "functional unit status" table that some instructions are not ready to be executed and so they are piling up in the scoreboard. These instructions include all of the instructions with Rj or Rk = No. This means that their operands are not ready and therefore the instructions are just waiting. They are detected by the values of Rj and Rk, which would likely be a value of 0 (for No) and 1 (for Yes).

47. Referring to claim 30, Stark has taught a processor as described in claim 23. Stark has not taught that the select loop generates a collision victim vector to identify collision victims. However, Hennessy has taught the concept of generating a vector within a scoreboard which is used to determine a collision has occurred. See page 247 and note that the SUBD and ADDD instructions collide because they both require the same functional unit (see caption). Since the SUBD is older and also the parent of the ADDD, it will be issued first, thereby generating the vector associated with the SUBD instruction. The vector comprises bits which represent the fields shown in the "functional unit status" and "instruction status" table. In this situation, the important data is that which specifies the ADD unit (which is used to execute the SUBD and ADDD instructions) is busy. By saying it is busy, the ADDD cannot wakeup and be issued for execution (as shown in the "instruction status" table). The bits used to specify the ADDD instruction has not issued indicates that it has collided with another instruction. A scoreboard

allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. See page 242. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard which is used to detect collision victims.

48. Referring to claim 31, Stark in view of Hennessy has taught a processor as described in claim 30. Hennessy has further taught that the collision victim vector is communicated to the wakeup loop. Again, see page 247. More specifically, this vector is used to direct wakeup of instructions. For instance, when the SUBD instruction is done with the ADD unit, then it will no longer be busy and therefore, the ADDD instruction may be woken up and issued.

49. Referring to claim 35, Stark has taught a method as described in claim 34. Stark has not taught that handling collisions includes generating a collision victims vector. However, Hennessy has taught the concept of generating a vector within a scoreboard which is used to determine a collision has occurred. See page 247 and note that the SUBD and ADDD instructions collide because they both require the same functional unit (see caption). Since the SUBD is older and also the parent of the ADDD, it will be issued first, thereby generating the vector associated with the SUBD instruction. The vector comprises bits which represent the fields shown in the "functional unit status" and "instruction status" table. In this situation, the important data is that which specifies the ADD unit (which is used to execute the SUBD and ADDD instructions) is busy. By saying it is busy, the ADDD cannot wakeup and be issued for execution (as shown in the "instruction status" table). The bits used to specify the ADDD instruction has not issued indicates that it has collided with another instruction. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their

operands are available. See page 242. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard which is used to detect collision victims.

50. Referring to claim 36, Stark has taught a method as described in claim 34. Stark has not taught that handling collisions includes generating a pileup victims vector. However, Hennessy has taught the concept of generating a vector within a scoreboard which is used to determine a pileup. See page 247 and note that the SUBD and ADDD instructions collide because they both require the same functional unit (see caption). Since the SUBD is older and also the parent of the ADDD, it will be issued first, thereby generating the vector associated with the SUBD instruction. The ADDD, on the other hand, must wait until the SUBD instruction finishes with the execution unit, and therefore it is a pileup victim (i.e., there is a pile of instructions waiting to use the ADD unit). The pileup vector comprises bits which represent the fields shown in the "instruction status" table. In this situation, the important data is that which specifies that the ADDD instruction has not issued. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. See page 242. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard which is used to detect collision victims.

51. Referring to claim 37, Stark in view of Hennessy has taught a method as described in claim 36. Hennessy has further taught that generating a pileup victims vector includes reading a scoreboard and identifying pileup victims based upon the scoreboard. See page 247.

52. Referring to claim 43, Stark has taught a method as described in claim 42. Stark has not taught that handling collisions includes generating a collision victims vector. However, Hennessy has taught the concept of generating a vector within a scoreboard which is used to determine a collision has occurred. See page 247 and note that the SUBD and ADDD instructions collide because they both require the same functional unit (see caption). Since the SUBD is older and also the parent of the ADDD, it will be issued first, thereby generating the vector associated with the SUBD instruction. The vector comprises bits which represent the fields shown in the "functional unit status" and "instruction status" table. In this situation, the important data is that which specifies the ADD unit (which is used to execute the SUBD and ADDD instructions) is busy. By saying it is busy, the ADDD cannot wakeup and be issued for execution (as shown in the "instruction status" table). The bits used to specify the ADDD instruction has not issued indicates that it has collided with another instruction. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. See page 242. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard which is used to detect collision victims.

53. Referring to claim 44, Stark has taught a method as described in claim 42. Stark has not taught that handling collisions includes generating a pileup victims vector. However, Hennessy has taught the concept of generating a vector within a scoreboard which is used to determine a pileup. See page 247 and note that the SUBD and ADDD instructions collide because they both require the same functional unit (see caption). Since the SUBD is older and also the parent of the ADDD, it will be issued first, thereby generating the vector associated with the SUBD

instruction. The ADDD, on the other hand, must wait until the SUBD instruction finishes with the execution unit, and therefore it is a pileup victim (i.e., there is a pile of instructions waiting to use the ADD unit). The pileup vector comprises bits which represent the fields shown in the "instruction status" table. In this situation, the important data is that which specifies that the ADDD instruction has not issued. A scoreboard allows instructions to execute out-of-order which allows instructions to execute as soon as their operands are available. See page 242. Consequently, less stalling is required. As a result it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Stark to include a scoreboard which is used to detect collision victims.

54. Referring to claim 45, Stark in view of Hennessy has taught a method as described in claim 44. Hennessy has further taught that generating a pileup victims vector includes reading a scoreboard and identifying pileup victims based upon the scoreboard. See page 247.

Response to Arguments

55. Applicant's arguments filed on December 27, 2004, have been fully considered but they are not persuasive.

56. Applicant argues the novelty/rejection of the independent claims on pages 25-27 of the remarks, in substance that:

> "In contrast, the cited portions of Stark describe conventional scheduling in which dependent instructions are not woken up until their parent instructions are confirmed as selected for execution."
>
> "even this second processor described in Stark requires an instruction to be confirmed as selected before it can trigger wakeup for a child or grandchild instruction. With respect to independent claim 15, the art of record fails to teach or suggest "a select-free scheduler", as claimed. The schedulers described in Stark require that instruction selection occur in order to trigger instruction wakeup. These schedulers cannot be considered select-free."

57. These arguments are not found persuasive for the following reasons:

a) The examiner asserts that applicant is reading the amended portions of the claims too narrowly. That is, the amended portions of the independent claims, when interpreted as broad as possible, still do not overcome Stark. More specifically, applicant, in essence, claims that dependent instruction wakeup occurs before selection is confirmed for the instruction causing the wakeup. Looking at Fig.11, for instance, it should be noted that any pipeline stage beyond the "select" stage confirms that the instruction has been selected. For instance, one could ask, "When the OR instruction of Fig.11 is in the execute stage in cycle 3, has the OR instruction been selected?" The answer is yes and the fact that the OR instruction is in the execute stage confirms that it has been selected; otherwise it wouldn't be in the execute stage. And, since the SUB instruction is woken up before cycle 3, the wakeup happens before selection is confirmed. This interpretation in itself is enough to anticipate the claims.

b) As for applicant's use of the term "select-free" scheduler, the examiner asserts that this is merely a label for a scheduler. Applicant has not defined, in the claims, what is meant by "select-free". Furthermore, claim 15 discloses select circuitry, which makes it sound as if the scheduler is not select-free. Clearly, some selecting is occurring. Applicant should elaborate on what is meant by select free.

Conclusion

58. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH
David J. Huisman
February 1, 2005

Eddie Chan

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100